A Preliminary Proposal for the IceCube DAQ System Architecture

Authors: C. McParland, J. Jacobsen, G. Przybylski, D. Nygren
Lawrence Berkeley National Laboratory

Revisions

- March 6, 2001 First draft (McP)
- March 15, 2001 Revisions (McP)
- March 16, 2001 Revisions (McP, Jacobsen)

Overview

This document is intended to describe an architecture for data acquisition and data handling for the Ice Cube experiment. It primarily covers the systems and components present at the experiment site (i.e. the South Pole). While it is intended to provide as much information as possible, many details will only be decided during the process of final system design. In its present state, this document is primarily an architectural description that should ***motivate further detailed discussions and design***. ***It does not yet contain sufficient detail to allow the task of task partitioning and implementation to begin***; but, it is a good first step. It should also be pointed out that this is not a requirements document. Once sufficient agreement has been reached on the system level issues described here, a requirements document should be generated. Such a document, and the mutual agreements it represents, provides the real description of how portions of the system must behave and interact with each other.

Several features are notable about this design. One is the prevalence of off-the-shelf computing and networking hardware from the earliest possible point in the data flow diagram. It should be further noted that the focus is on commercial hardware and software found in the world of e-commerce and Web services industry as opposed to process control. Furthermore, there is a clear emphasis on the use of commercial Ethernet networking protocols and hardware in place of more traditional bus-based architectures such as VME or CPCI. Both of these features are intentional and represent our belief that, given both the expected data rates and the difficult maintenance environment at the South Pole, these choices represent the best solutions for both system performance and stability.

Table of Contents

Instrumentation and System Performance Goals

The following is a summary of the expected data rates from digital optical modules in the ice operating under normal conditions. It also describes system performance goals based on expected input data rates. Rates and performance goals are based on instrument physics requirements and data collection targets described in **OTHER DOCUMENTS**. All descriptions and discussions of these matters are referred to these sources.

Detailed information and rate calculations can be found in the Appendix.

A.1. DOM Data Rates

During normal operations, we expect each DOM to emit data describing approximately 500 hits/sec. Given an assumption that 95% of these hits are easily characterized SPE waveforms and assuming some degree of data compression on the remaining complex waveforms, we expect a data rate of approximately 10kB/sec. per DOM. This bandwidth is divided roughly equally between single photoelectron hits and complex waveforms.

A.2. String Data Rates

Since DOMs are currently configured to share a single twisted pair, we will see roughly twice that data rate on each twisted pair or about 20 kB/sec. per DOM pair. (Note that current measurements indicate that the maximum data rate achievable for a single twisted-pair cable is approximately 50kB/sec - this presents the tightest constraint on bandwidth in our design.) Given a total of 60 DOMs per string, one has approx. 600kB/sec. per string.

A.3. Global Trigger and Event Builder Rates

Assuming a coincidence rate of about 60/sec. per string and four hits per coincidence, we expect a rate of about 2kB/sec. between each string processor and the global trigger or a total of 180kB/sec. for the entire array. Return messages from the global trigger to each string processor add an additional 7kB/sec. per string or about 580kB/sec. for the array. Actual string data transferred to an event builder constitutes approximately 800kB/sec. for the entire detector array.

B. Information Capture in a Digital Optical Module (DOM)

- **PMT Waveform Data**

Primary information capture occurs within the DOM by triggered acquisition of PMT anode current waveforms using the Analog Transient Waveform Digitizer (ATWD) ASIC. The threshold for triggering is set to the lowest practical value that responds to most of the SPE signals, but rejects the vast majority of non-photoelectron pulses arising from electronic noise or PMT dark current. The waveform capture is sufficiently detailed to resolve nearly overlapping signals. This implies a sampling rate in the range of 250 – 320 MHz for the PMT characteristic pulse shape. The highest practical dynamic range is desired consistent with the maximum output of the PMT. The three input channels of the ATWD permit this ASIC to provide the rough equivalent of a 14-bit ADC. The 128 samples/channel provide a time span for high-resolution waveform capture of 400 ns at 320 megasamples per second. The digitized waveform is typically compressed and forwarded in digital form to the data acquisition hardware. In addition, 10-bit 33

MHz FADC data stream arranged with an appropriate shaping time provides a continuous record of event structure, with ~30 ns resolution; for most events, this data stream is ignored.

- **Time-stamp Process**

The triggering of waveform acquisition is time-stamped according to a stable local clock within the DOM (whose frequency for the currently deployed DOM prototype string is f = 33.6 MHz). The time-stamp is a 32-bit integer representing local time in units of 1/f. The local clock and, thus the recorded waveform timestamp, are calibrated against a time source located on the surface. (The time source for the prototype string is a GPS clock, although experience is showing that a cesium or rubidium clock may be more suitable.) When proper time base corrections are applied for each individual DOM, the resulting waveforms for all DOMs are synchronized to a common time source.

- **Analog – Digital Conversion Process**

Every captured analog waveform is digitized with 10-bit resolution using the ATWD's internal common-ramp Wilkinson ADC. Counting occurs on both positive- and negative-going edges. Conversion takes 512 x (1/f) = 15.36 microseconds at 33.6 MHz. Further processing may include digitization of additional ATWD channels if the low gain range shows saturation in the data. 14 bit dynamic range is accomplished by capturing the waveform simultaneously in multiple channels at different gains, then using the channel with the highest gain which isn't clipped.

- **Feature Extraction Process**

The vast majority of digitized signals are single photo-electron (SPE) pulses. The information content of these waveforms consists primarily in the timing, and secondarily in charge. These signals can be fully compressed into a small 8 to 10 byte packet that contains the magnitude and timestamp for each signal. Some portion of the digitized signals (5 to 10%) will contain complex waveforms representing overlapping waveforms from multiple pulses. These signals must be characterized and passed to the surface in greater detail. The exact meaning of the accompanying timestamp for these complex waveforms will remain ambiguous until analyzed further. However, the timestamp corresponding to the waveforms first excursion over threshold is included for later timing calculations.

- **Data Acquisition**

The above sequence constitutes the detector's real-time information capture cycle. From this point on, data flows from the DOM to the data acquisition system where it is analyzed for both time and spatial correlations with data from other DOMs. After digitization and feature extraction in the DOM, the subsequent tasks are essentially one of data communications and processing. The DAQ system is responsible for producing streams of correlated data for further analysis while discarding the overwhelming majority of useless background data being generated.

C. System Overview and Network Implementation

C.1. Overview of Components

The system consists of an interconnected set of hardware components and software tasks. While this design does allow for a great deal of flexibility in how these components are grouped together and mapped onto various networks and processors, the overall system performance goals dictate that certain aspects of this grouping are fixed. In the following discussion, the distinction between system software processes and actual CPUs within the system should be noted. For example, a complete string of optical modules is said to terminate in a single processor, called the string processor. For the purposes of system design, it has been important to decide that, since an optical module string represents a logical grouping of data for triggering purposes, all data from one string will always flow to a single processor and will not be distributed across multiple CPU's. While, for the purposes of this document, it is easiest to discuss the string processing program as, in fact, executing on a single processor, namely the string processor, the system design does not require that each string be handled by a separate processor . It does, however, require that all data from all DOMs in a string are routed to a single system process that can deal with them in an efficient and complete manner.

C.2. Network Topology

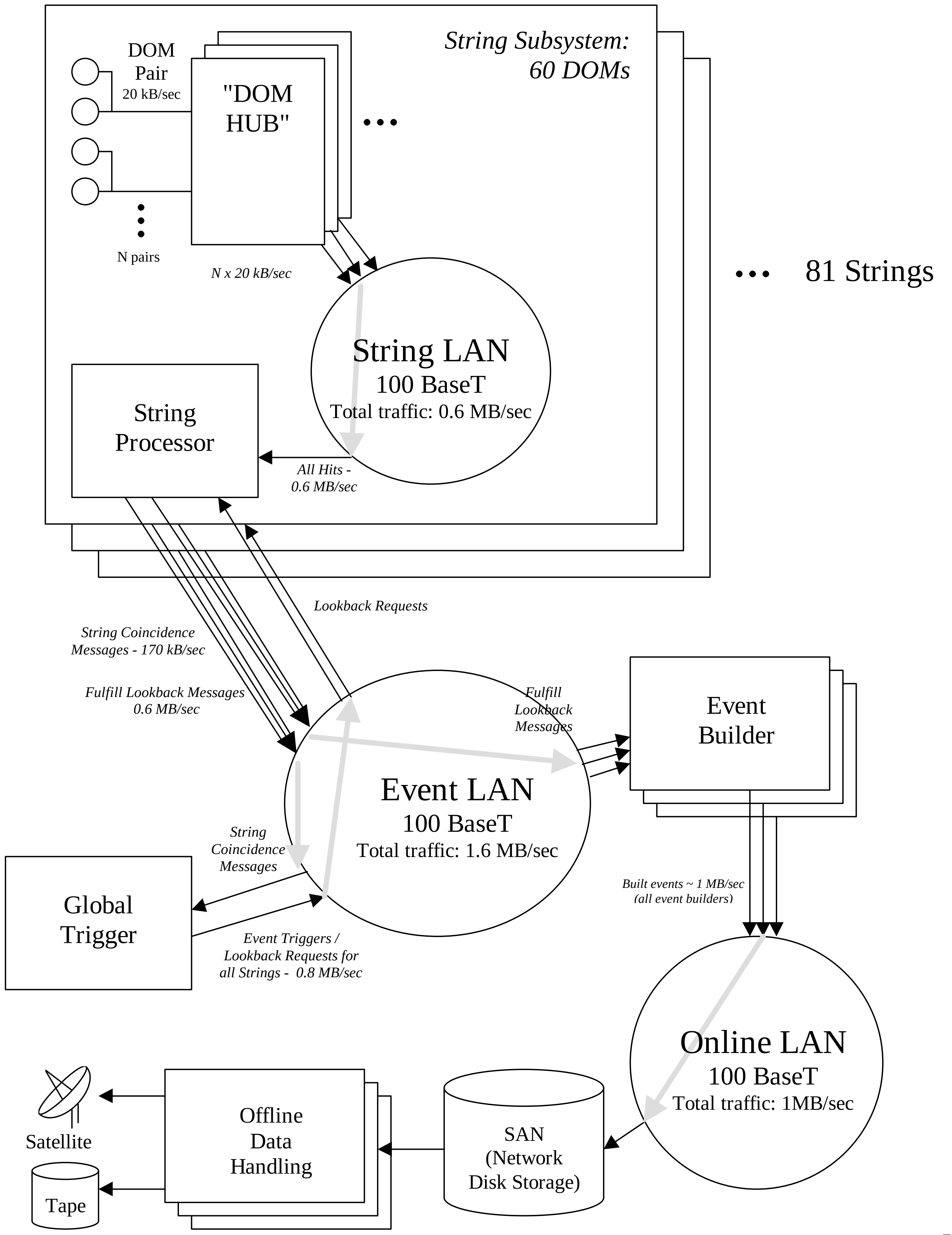
We begin with a high level system block diagram. The following figure shows the major components of the system architecture as well as their network communications paths.

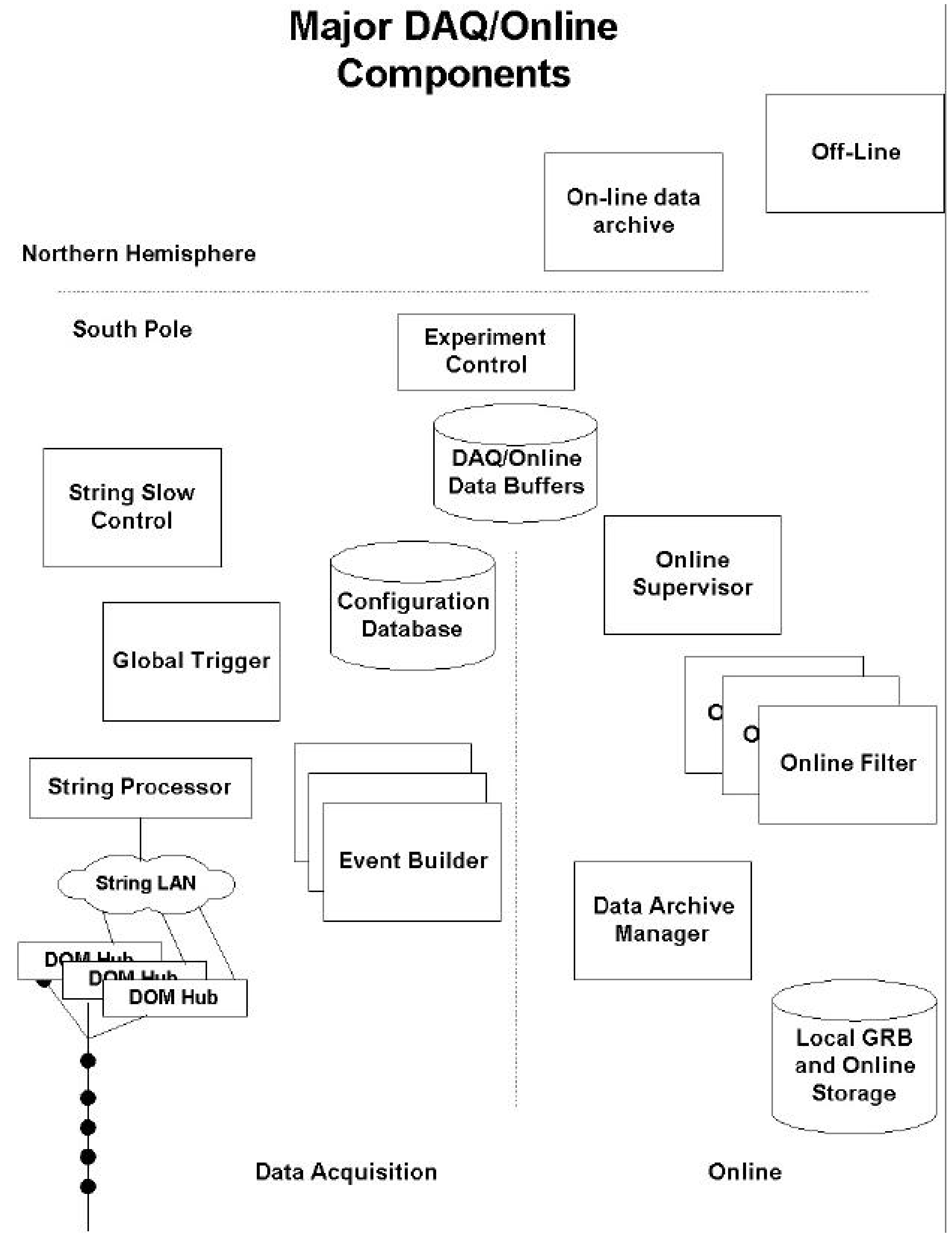
C.3. Relationship of Major DAQ/Online Components

Ice Cube software architecture is divided into three major partitions: data acquisition, online processing and monitoring, and off-line analysis. The data acquisition partition is wholly located at the South Pole and is primarily responsible for interacting with detector electronics and DOM readout. Online filtering and monitoring is primarily located at the South Pole and is responsible for event filtering, event recording and GRB search functions. It is also responsible for data transfer between the South Pole and the designated data archiving site (e.g. NERSC). Offline analysis is responsible for final Ice Cube data reduction and analysis.

At the present stage of system design, it is important to understand the ways in which these elements need to interact and to sufficiently analyze these communications requirements to assure their ultimate implementation. The following diagram lists major components of this architecture and shows, to some extent, their placement and relationship to other components.

Proposed IceCube DAQ Network Architecture


String Subsystem: 60 DOMs
DOM Pair 20 kB/sec
"DOM HUB"
N pairs
N x 20 kB/sec
81 Strings
String LAN
100 BaseT
Total traffic: 0.6 MB/sec
String Processor
All Hits - 0.6 MB/sec
Lookback Requests
String Coincidence Messages - 170 kB/sec
Fulfill Lookback Messages 0.6 MB/sec
Fulfill Lookback Messages
Event Builder
Event LAN
100 BaseT
Total traffic: 1.6 MB/sec
String Coincidence Messages
Global Trigger
Event Triggers / Lookback Requests for all Strings - 0.8 MB/sec
Built events ~ 1 MB/sec (all event builders)
Online LAN
100 BaseT
Total traffic: 1MB/sec
SAN (Network Disk Storage)
Offline Data Handling
Satellite
Tape


Major DAQ/Online Components
Off-Line
On-line data archive
Northern Hemisphere
South Pole
Experiment Control
DAQ/Online Data Buffers
String Slow Control
Online Supervisor
Configuration Database
Global Trigger
Online Filter
String Processor
Event Builder
String LAN
Data Archive Manager
DOM Hub
DOM Hub
DOM Hub
Local GRB and Online Storage
Data Acquisition
Online

D. Overview of Data Flow in IceCube DAQ and Online

The proposed DAQ design revolves around three separate phases of event creation:

- Within a string, the collection of hits into "string coincidence events."
- The collection of string coincidences and the decision to form a global event trigger.
- The gathering of hits on all strings which belong to global events, and the recording of the complete event information on disk and tape.

These tasks can be laid out in more detail by enumerating the messages which are exchanged on the various networks of the DAQ. The following items are also summarized in the diagram in the above figure. Since 100BaseT networking is assumed, bandwidth requirements are shown to be comfortably less than 8 MB/sec in each case.

For each string, **hits from DOMs connected to DOM Hubs are sent over a String LAN to a String Processor.** There are several DOM Hubs, one 100BaseT network switch, and one String Processor per string. Assuming a noise rate of 500 Hz, 60 DOMs per string, a fraction of 5% of hits needing full waveforms, 208 bytes per waveform and 9 bytes per non-waveform hit, one obtains a data rate of about 0.6 MB/sec per string.

Each String Processor is also connected to a global Event LAN**. String coincidences are reported by each String Processor to the Global Event Trigger** on the Event LAN. The Global Event Trigger is envisioned as a single processor on the Event LAN, which is also connected to a set of processors that serve as event builders. Assuming a string coincidence rate of 60 Hz per string, an average of 4 hits per coincidence event, and 60 DOMs per string, one obtains a total data rate of less than 0.2 MB/sec due to these messages.

When conditions for a particular trigger type are fulfilled, **the global event trigger informs the String Processor to tag all hits which occurred near in time to the global trigger.** If one assumes a global aggregate trigger rate for all trigger types of 1000 Hz, 10 bytes per message, and 81 string processors, these "Lookback Requests" require an average bandwidth of 0.8 MB/sec.

The String Processor sends the data for the tagged hits to one of a set of Event Builder CPUs. This also occurs over the Event LAN. Assuming 20 random noise hits per event, 12 muon hits per event, and the above values for the average number of bytes per hit and for the global event rate, one obtains an average bandwidth of 0.6 MB/sec. This brings the total bandwidth usage on the Event LAN to about 1.6 MB/sec, well within the capabilities of 100BaseT switches.

The Event Builder CPUs send fully constructed events to a disk server over the Online LAN. The disk server will write events to disk. Since the bulk of the event data is in the hits, the bandwidth on this network will be not significantly more than 0.6 MB/sec. 1 MB/sec is a safe number to assume, well within the 100BaseT limits. In addition to the event builder processors and the disk server, the Online LAN also is connected to the processors that constitute the online filtering and data archiving system.

This design is attractive, in part, to its high degree of modularity and flexibility. Thus to create a minimal test stand for DOM production and quality control, one can simply work with a single DOM Hub attached to one or more DOMs (or DOM pairs). Functionality for a whole string can be achieved by networking the DOM Hubs to an off-the-shelf String Processor CPU and installing the requisite software. In an international collaboration where hardware production, software development, testing and deployment will be carried out on multiple continents, this is a key advantage. Finally, with the proposed "network-heavy" scheme, we benefit from a tremendous amount of community experience and publicly-available tools for TCP and UDP-based networking, which far outstrips the resources available for bus-based architectures such as VME or Compact PCI.

E. Digital Optical Module (in Ice)

E.1. DOM Architecture

A detailed description of the DOM design is outside the scope of this document. However, it should be pointed out that the DOM is a self contained data acquisition platform that is capable of digitizing real-time PMT pulses, storing them internally, and, when requested, transmitting them to surface electronics. It contains its own processor, memory, and flash file system. It executes a real-time OS and is capable of scheduling background communications tasks while acquiring data from a local waveform digitizer. For the purposes of this document, several important DOM operations will be described in greater detail.

E.2. Slow Control and Self Test Features

Each DOM is responsible for controlling critical data taking parameters. These include setting local PMT high voltage levels, discriminator threshold levels, ATWD digitizer control voltages, and even selection and loading of appropriate FPGA programs. Since none of these operations is performed during actual data acquisition, they are deemed part of the slow control system. All adjustments capable of affecting the quality and quantity of data produced by the DOM are controlled from within the DOM itself. Therefore, all slow control requests, regardless of their origin (user interface, automated experiment control sequences, etc.) are finally serviced within the DOM and its resident software.

Each DOM is also capable of carrying out extensive self test sequences. In addition to verifying correct operation of the digital platform and peripherals within the DOM, these programs can test analog electronics for increased noise levels or spurious discriminator or ATWD behavior. Upon initiation, these tests are carried out under the control of the local DOM processor and results are reported back to surface electronics upon their completion.

E.3. Waveform Digitization and Time Stamping

The DOM captures waveforms for each PMT “hit”, where a hit is defined as a pulse that exceeds a discriminator threshold held at a small fraction of a single photo-electron (SPE). The waveforms provide accurate information about the SPE pulse shape and its spectrum, which is essential for the determination of the PMT SPE efficiency. The four channels of the ATWD can provide the rough equivalent of a 14-bit ADC, with sample speeds up to ~1 GHz. An active delay line controls the sampling action; no high-speed clocks or high-power dissipation circuit elements are needed. A single DC control current varies the sampling speed. In addition, a PMT signal is generated with ~30 nsec shaping, and digitized by a dedicated 10-bit FADC (operated at 33.6 MHz in the DOM prototype string). This information channel can be used to obtain records longer than the ATWD span, such as may occur during extremely energetic shower events.

In addition to PMT signal waveform capture, the DOM also timestamps each PMT hit to the required accuracy and resolution. This requires that an appropriate timing signal be available from a local clock in the DOM, which is calibrated (in frequency and offset) against a master clock on the surface. The method adopted for the realization of this capability is perhaps the most unusual feature of the DOM concept. Though unorthodox in concept, the circuit design, components, and operation are completely conventional.

E.4. Feature Extraction and Data Flow

The DOM must determine which signals should be recorded with their accompanying waveform. This is required in order to keep the data rates to the surface at reasonable levels. The provision of a robust algorithm for this purpose is an essential part of the DOM functionality. The total data flow from a typical DOM is therefore expected to be ~10 Kbytes/second. If the PMT hit rates are more like the expected <500 Hz, then the data rates would be proportionately smaller. Since there will be two DOMs per twisted pair, the rates must be doubled to obtain the total upward data traffic on each twisted pair.

Data from each DOM is requested periodically, perhaps at a rate of ~10 Hz. This provides adequately small latency for data streaming purposes, and maintains a reasonable transmission burst of about 1 Kbytes/request. Each request for data includes the time-mark signal. The DOM will ignore most of these, using them only as frequently as local clock drift characteristics require.

F. Digital Optical Module (surface systems)

F.1. DOM Hub Architecture

The current surface electronics package is based on an embedded processor design that provides low level communications services between the surface and the DOM and provides high level communications services (i.e. TCP/IP) to the string processor. This component, the DOM Hub, will consist of 1U rack-mounted chassis with front end electronics for 4 or 8 DOMs, an embedded CPU daughter card for processing and communications functions, and a power supply capable of providing power for all attached DOMs. The back panel of the chassis contain connectors for direct attachment to DOM deployment cables and a standard 10/100BT network connector. The embedded processor is a sub credit card size 200 MHz StrongARM CPU running Linux.

F.2. DOM/Surface Time Synchronization and Control

Synchronization between each DOM and surface systems is accomplished by sending an accurately timed pulse from the DOM Hub to the DOM. Since this pulse must not be confused with normal communications signals, the DOM Hub is responsible for notifying the DOM operating system that synchronization is to take place, emitting the GPS-synchronized signal on the DOM cable, and then restoring normal DOM/surface communications. Surface to DOM cable delay is determined by sending a timing signal from the surface to the DOM and, after a known delay, sending an identical signal from the DOM to surface. If the circuits generating and receiving these signals are identical, and the analysis of the signals (e. g., for a zero crossing) are identical on surface and in the DOM, then the cable delay is simply one half the measured sum of the propagation times in each direction, after subtracting the programmed delay in the DOM.

In practice, the DOM must receive, time-stamp and digitize a typical time-mark signal. Having processed such a signal, the DOM, after waiting a predetermined number of local clock cycles to let the cable become quiet, transmits an identical time-mark signal back up the cable. At the surface, the front-end circuitry captures, time-stamps, and digitizes the return time-mark signal generated by the DOM. As mentioned above, it is importantthat the surface front-end circuitry be identical to that in the DOM for this function to display the needed symmetry. The zero-crossing times of the time-mark signals are calculated from the digitized DOM and surface time-mark waveforms. In essence, the arrival times of successive pulses that are sent in the same direction determine the clock frequency; the round-trip time determines the clock offset.

This “**Reciprocal Active Pulsing**” or RAP method has the additional important advantage in that the procedure can be initiated under software control, and requires only a small fraction of a second to complete. The procedure may be easily repeated as many times as needed to ensure reliability and precision. Commissioning and monitoring are facilitated by this capability.

F.3. DOM Data Stream Handling

The DOM Hub will communicate with computers on the String LAN through a 10/100-Base-T network interface. The DOM Hub will support TCP/IP protocol and will deliver data streams for each DOM to the string processor using the standard socket programming model. Furthermore, the DOM Hub will provide transparent Web-based communications with individual DOMs so that maintenance and minimal testing functions may be performed remotely without the need of the full data acquisition system.

G. Experiment and Configuration Control

The system design described in this document is based on a loosely-coupled collection of sub-systems whose operations are, for the most part decoupled. One advantage of such a design is that it allows some independence in the design and implementation of individual sub-systems. However, in order to co-ordinate these elements into a single, coherent system, some degree of overall control is required. This function is provided by the experiment control system

The Ice Cube detector can, at any given moment, be described as being in one of several operational states. These states serve as short, meaningful descriptions of just what operations are being performed by the detector. By enumerating which operations are allowed in which detector states, this state model also serve as a mechanism to control and co-ordinate operations of separate detector sub-systems.

G.1. Scope of Experiment Control

The detector, as a whole, can be though of as being in one of the following states: "idle", "entering data-taking state", "running", "paused", "exiting data-taking state", "calibrating" or in one of several "testing" states. Each of these states has certain implications for the operations of each sub-system. For example, if the detector is taking data, the DOM string control sub-system should not be able to alter PMT high voltage settings on an active OM. Similarly, although it is permissible to alter individual settings for an inactive DOM (i.e. an DOM not included in the global trigger's list of active DOMs), it should not be possible to initiate LED pulsers on this particular DOM since resulting light would affect data taking activities. Therefore, all sub-systems refer to experiment control as the single indicator of the system's current aggregate operational state. Each sub-system is responsible for designing its internal operations and states to be consistent with the overall detector behavior described by experiment control's current state.

G.2. Common Component Control Model

In addition to providing all sub-systems with a consistent indication of the detector's overall state, experiment control must, in order to initiate and control data taking, be able to request sub-systems to perform certain actions. Depending on a particular sub-system's role in the detector's operation, these actions can be relatively simple or fairly complex. For example, when beginning data taking, DOM string control will be required to log current DOM parameter settings to a local database and change its state from "idle" to "data taking". The fact that the DOM string control sub-system is in the "data taking" mode implies that it will be unresponsive to user requests to reset critical DOM parameters. Other sub-systems may require more complex responses to experiment control commands.

To the extent that the complexity of these operations is only of concern to the sub-system itself, the details of these operations should remain concealed within the sub-system's operation. Each sub-system should be able to accept a high-level request from experiment control to move into a new state. Following this request, each sub-system will respond with its new state and status. Successful transitions of each sub-system into the appropriate state will lead experiment control to indicate that the detector has taken on the desired final state (e.g. "running"). The failure of any sub-system to appropriately move to the desired target state will indicate a system problem and will cause experiment control to bring all sub-systems back to a consistent and safe state.

It should be noted that with the loosely-coupled system described here, individual control operations are best handled directly by those sub-systems involved. That is, commands and requests for monitoring and status information will be processed by the sub-systems themselves, not by some single intervening agent (e.g. experiment control). The advantage of such a design is that additions and alterations to sub-system controls and interfaces need only be implemented in one place -the sub-system itself. They do not have to be further integrated into additional supervisory programs or levels.

However, it should be pointed out that, with this implementation model, it is possible for there to be a wide and confusing variety of user interfaces for each sub-system. Even with experiment control properly playing its detector-wide coordinating role, it will be possible to implement sub-system user interfaces in many different and potentially awkward styles. A certain degree of discipline in design and implementation of individual sub-system user interfaces will be required in order to produce a consistent overall look and feel. Given the breadth of functions requiring user interfaces and the necessity for effective remote (albeit secure) access, the adoption of a Web-based user interface model is preferred.

G.3. DAQ Configuration Database

The DOM string control sub-system will be responsible for keeping an accurate database of DOM-related parameters for each data taking session. While this list of DOM parameters and operational settings is not yet complete, it is safe to assume that, during any given data taking session, each DOM participating in the detector array will have 50 to 100 parameters that describe its intended operation. These will include static fiducial information such as serial number, string and position location, firmware version number, etc. It will also include per session information such as PMT high voltage, trigger threshold level, and, if some level of waveform capture is included in the DOM, parameters that describe operation of waveform capture and feature extraction algorithms.

This same database will also be used to control which DOMs are actually participating in data acquisition operations for detector array and, if required, insure that unused DOMs are powered down and/or not contributing unwanted triggers to string processor trigger processors. This same database will be used by the global trigger and event builder sub-system to control operations of the string trigger processor algorithm.

G.4. Component Discovery and Self Configuration

The Ice Cube detector will be deployed over a period of 5 to 6 years. Furthermore, during the summer deployment season, optical module strings will be deployed, tested and integrated at rates far in excess of that experienced by the current collaboration. While the physical logistics for drilling, deploying and cabling detector strings are critical to the success of this timetable, the speed with which the data acquisition system can adapt and re-configure to the addition or deletion of new optical module strings will be of equal importance.

Each sub-system located at the South Pole must be capable of interrogating its environment and determining whether needed hardware and software components are available. This operation can be performed automatically when individual sub-systems are started or can be the result of an explicit re-configuration command. The most basic example of such an operation is enabling the experiment control sub-system to locate, via a set well-defined network operations, and establish communications with all other operating sub-systems. A more complex, but conceptually similar, configuration operation will enables the DOM string control sub-system to determine which strings are present and which optical modules are in an operable state. This resource discovery scheme can be extended to on-line sub-systems as well by enabling the filter farm manager to discover which individual processors are available to participate in its operation.

If this technique is consistently applied across all sub-systems and is combined with accurate and complete documentation of the experiment cable plant, it becomes relatively simple to incorporate both additional DOM strings and computing resources as they are installed. The presence or absence of portions of the hardware and software system become apparent to operators as each sub-system reports its success or failure during a "configure" operation. In addition to speeding integration of new portions of the detector, this capability allows fast and effective resolution of complex system-wide equipment problems. This capability can also provide critical equipment status information when recovering from a power black out- not an unheard of occurrence at the South Pole.

Just as it is critical to be able to automatically incorporate new DOM strings and computing components as they are installed, it is just as important to be able, under computer control, to mark portions of the system as off-line or "non-participating". The most obvious need for such a facility is during system debugging. It is often useful to remove elements from the system in order to investigate their role in generating specific error conditions. Unfortunately, it is not always convenient or possible to electrically disconnect or power down suspected hardware or software components. Some data acquisition problems may require this sort of system de-population for correct diagnosis and repair. If these problems occur during the winter season, the ability to remove some components from the system, via remote access, will be of great advantage.

G.5. Security Environment

"Security" can cover a wide range of capabilities and protections. It is worth reviewing the current use of the term in computing and communications systems. In current computing usage, the term security is based on four principles: authentication, authorization, privacy, and non-repudiation. Briefly, authentication involves the verification that a user has the identity they claim to have, authorization verifies that a given user has the authorization or privilege to perform a requested operation, privacy assures users that messages and data exchanged with another user will be private to only those two parties, and non-repudiation stipulates that requests received from a user cannot be repudiated at a later date (e.g. electronic transfer of funds). Each of these principle functions can be implemented separately and each with varying degrees of completeness. Furthermore, several different technologies can be used to achieve some or all of this functionality. While simple open password schemes are not considered to provide any real level of security, suitable software mechanisms, such as DES, Kerberos, and PKI (Public Key Infrastructure), exist and are readily available.

An exact statement of Ice Cube security requirements does not, as yet, exist. However, based on past Amanda experience and reasonable projections of Ice Cube operations, several needs are obvious. Regardless of what network access level protections may be implemented upstream, such as screening of requestor IP addresses, it is important that all control commands be verified and screened by a security service at the South Pole site. This service should provide, at a minimum, strong authentication and authorization functions. These would basically regulate who would have access to data acquisition and on-line control and monitoring information and which specific capabilities would be allowed on an individual basis. While privacy and non-repudiation functions do not appear to have any role in experiment control and monitoring at present, any suitable security service will allow their implementation at a later date.

In discussing security in the context of experiment control, it is assumed that a certain level of robust security is provided by the underlying operating system and network environment. It is assumed, for example that unauthorized access to system accounts has been eliminated and that network routers and their control ports have been made suitably safe. The following discussion centers on the application design decisions that will enhance this low-level security and protect the resulting system from unauthorized or inadvertent use.

One mechanism used in designing secure control systems is to create a finite number of "roles" that users can take on when logged onto a system. These roles are typically descriptive in nature (e.g. "operator", "implementers", "maintenance technician", "super user/god"). Users requesting access to the data acquisition or on-line sub-systems at the South Pole will present their name, security credentials, and intended operational role to the security service. After authenticating user identity, the security service determines whether a user is authorized to take on the requested operational role. If these security checks are successful, the user then has all the privileges associated with that operational role.

One of the advantages of this scheme is that it creates a clean partition between security services and system operations. Security mechanisms can be centralized into a single point of access. Individual sub-systems are then able to treat all commands as having been issued by authenticated users acting in an authorized capacity (i.e. role). In addition to these implementation efficiencies, this scheme also clarifies overall system design and operation. Although commands to sub-systems will carry the identity, as well as the acting role, of the requestor, the design becomes one primarily driven by the requestor's role. The act of describing the set of all possible user operational roles implicitly describes all possible modes of system operation. For example, since each sub-system differentiates between normal operations allowed by an "operator" and those privileged commands reserved for "implementers", lists of permissible "operator" and "implementer" functions take on detector-wide meaning and significance. Furthermore, since, at the sub-system implementation level, every command must be checked against the role of the requestor, sub-system behavior becomes both well-organized, easily described and, hopefully, documented.

H. DAQ Components

H.1. Slow Control

The DOM string control sub-system will consist of a central control program and co-operating tasks executing on each string's dedicated control processor. Since the detector will grow by the addition of

DOMs organized as strings and since the only electrical and logical path for communicating with a particular OM is through its string processor and cable, this design is a natural fit for the detector's topology.

The central slow control program will be responsible for all user and experiment control interface functions to the optical modules and will be responsible for all DOM-related interactions with the detector configuration database. These database interactions will include queries of DOMs currently active in the detector, setting and recording of internal DOM parameters for data taking, and recording of individual DOM calibration information acquired during low level calibration operations. Most DOM string control operations will be synchronous command or monitoring requests.

At present, there are only a small number of possible states for the DOM string control sub-system. These states should be sufficient to indicate the following DOM string control activities: off-line or idle, performing self-configuration and string discovery, performing one or more local DOM calibration functions, on-line and able to accept operator commands, and on-line but only accepting monitoring requests (as during normal data taking). Additional states can be added as needs arise. It should be noted that these states, as well as those of other sub-systems, will not correspond exactly to those of the experiment control program. They do, however, represent a list of possible sub-system behaviors that are consistent with the overall detector operation.

The central control program will also be responsible for discovering via their associated string processors, all DOMs presently in the detector. Once connected to the local area network, newly added strings will be incorporated into the central control program through a single "reconfigure" command. This same operation will allow strings temporarily removed from the detector for purposes of test or electronics repair to be marked as "non participating strings" and their respective DOMs recorded as "off line".

H.2. String Data Handling

Each DOM in a string produces a stream of hits and waveforms that, via the String LAN, ultimately reaches the string processor. All operations surrounding data transfer, sorting and interactions with other system components (e.g. the global trigger) are the responsibility of the string data handling process. The first phase of its operations center on the delivery of string level hit information to the global trigger process. Upon reception of data from one or more DOMs, its initial task to sort this data into structures that allow quick, efficient determination of local coincidence during a sliding time window (typically 8 usec.). When all coincidences for a given time period have been found, they are properly formatted and passed on to the global trigger process.

The second phase begins when the global trigger process responds with a list of time intervals of interest. It is then the string data handler's task to collect raw data from all DOMs for each of the time intervals of interest and forward the resulting data set to the appropriate event builder. In practice, these two phases will overlap and provision must be made to have several such operations in progress at any one time.

There are only a small number of possible states for the string handler process. These states should be sufficient to indicate the following activities: off-line or idle, performing self-configuration and DOM discovery, processing normal data triggers, paused due to operator request or error condition, and performing string specific tests. Additional states can be added as needs arise.

H.3. Global Trigger

As described above, the global trigger is responsible for generating lists indicating the occurrence of detector-wide events of interest. It consumes the list of hits as determined by the string data handling process in each string processor and emits a list of UTC-based times that bracket hits considered to be possible events. The global trigger behaves synchronously with the aggregate collection of on-line string processors. That is, the global trigger waits until all string processors that are configured and active to report their contributions for a given time interval. Only when all string contributions are present, can the global trigger begin the process of determining what events might have occurred. Once this determination is complete, the global trigger selects an event builder for the current data sequence and passes a list of UTC time intervals to both the selected event builder and all string handling processes.

During normal operation, all 81 string processors will remain synchronized within the reporting period specified when data taking was initiated. These periods will range from 200 msec. to several seconds, depending on DOM trigger rates and overall system performance. As described above, for each reporting period, ALL string processors must forward string-level local coincidences, or their absence, to the global trigger in order to initiate the global triggering process. In the absence of such a report, the global trigger will re-synchronize all string processors and forward, to the event builder an indication of what reporting periods were discarded. Persistent failure of a string processor program or CPU will cause the global trigger manager to enter a non-operating error state. Experiment control will detect, during periodic system scans, this state change and force an abnormal end of run sequence for all data acquisition sub-systems. Co-ordination activities and generation of global triggers are performed by a single global trigger process executing on a dedicated processor. Since this activity requires the exchange of multiple messages between this process and each string's data handling process during each reporting period, high speed, low latency network connections between these processors is of great importance.

There are only a small number of possible states for the global trigger sub-system. These states should be sufficient to indicate the following activities: off-line or idle, performing self-configuration and string processor discovery, processing normal data triggers, paused due to operator request or error condition, and performing global trigger specific tests. Additional states can be added as needs arise. As with other sub-systems, it should be noted that these states need not correspond exactly to those of the experiment control program. They do, however, represent a list of possible sub-system behaviors that are consistent with the overall detector operation.

H.4. Event Builder

The event builder is responsible for creating well-formatted, consistent data structures that contain contributions from any optical module for specific detector-wide times as determined by the global trigger manager. Once these data structures are complete, they are passed out of the data acquisition system partition and into the Online software partition where they undergo further filtering and, ultimately, archiving.

The current design does not anticipate any physics content in the operations of the event builder. Once a data stream has been selected by all participating string data handling processes and transferred to the selected event builder, the remaining task is primarily one of data movement and formatting. Since the global trigger process is capable of indicating a trigger type for each UTC time sequence of interest, the event builder will format an individual data stream consisting of time ordered contributions from each string for the time interval and trigger type of interest.

It should be noted that the resulting data streams from each event builder instance will cover different time intervals. As a result, the data for an entire run will be stripped across a sequence of files that each span a different time interval within the respective run. In putting together a time ordered data stream for archiving purposes, the Online system will need to properly order contributions from each of these files in order to obtain a well ordered output stream. Therefore, the event builder and Online system will share a simple data base that facilitates this process.

There are only a small number of possible states for the global trigger sub-system. These states should be sufficient to indicate the following activities: off-line or idle, performing self-configuration and string processor discovery, processing normal data triggers, paused due to operator request or error condition, and performing event builder specific tests. Additional states can be added as needs arise. As with other sub-systems, it should be noted that these states need not correspond exactly to those of the experiment control program. They do, however, represent a list of possible sub-system behaviors that are consistent with the overall detector operation.

I. Calibration Operations

Calibrations serve a crucial and central role in the operation of the IceCube detector. They must, to the greatest extent possible, be automated. The present digital design allows calibrations to be carried out without any physical intervention to the data acquisition system. That is, all calibrations can be performed without disconnecting cables or physically re-configuring DOM electrical data paths. However, while this is a necessary condition for fully automated, unattended calibration operations, it is not a sufficient one. Further design and detailed descriptions of operational sequences are still required for automated operations to become a reality. While the following framework should allow such operations to take place, further discussions are needed in order to insure their successful operations within the current design.

I.1. DAQ Calibration Operations

The primary calibration operation carried out within the DAQ system is that of synchronizing time bases for all DOMs with the GPS clock. This operation will take place automatically once every 5 to 10 seconds. The actual, ns. level signaling operations are performed by the DOM hub and DOMs themselves. The DAQ's only role is in enabling such operations and supervising the dissemination of the calculated clock offsets for each DOM to both the DAQ configuration data base and, through an as yet unspecified mechanism, the Online system. The primary DAQ component involved at the supervisory level is the string processor. Since all data leaving the string processor is corrected to UTC, this is the rational point at which to control and implement needed time base corrections. All other calibration IceCube calibration operations are based on analysis of data collected by a carefully configured, but normally operating DAQ. Therefore, these calibration operations fall outside the scope and capability of the DAQ and are, more properly, Online functions.

I.2. Online Calibration Operations

In addition to the time base synchronization calibration performed by the DAQ system, additional calibration operations are required for successful detector operation. These include, but are not limited to, determination of individual PMT high voltage settings, determination of proper PMT trigger thresholds and measurement of cable delays for all DOMs. As noted above, all of these calibration tasks are based on specific data analysis applied to the detector data stream. Therefore, these calibration operations are primarily operating in the Online domain. In fact, they can all be considered to be part of a special subset of normal data taking operations. It should be noted that some of these Online calibration operations will produce data required for correct operations within the DAQ system. In these cases, mechanisms must be in place for inserting these calibration values into the DAQ configuration data base.

In practice, these operations can be automated and orchestrated by the main supervisory program, experiment control. Experiment control can select a particular DAQ configuration and instructing DAQ to begin operations. It can then select a particular Online configuration and instruct Online to begin operation. With a suitable level of flexibility at the experiment control level, fairly complex sequences of calibrations can be carried out in an automated manner.

J. DAQ and Online Monitoring

J.1. Operations and Error Logging

The design described in this document partitions system behavior into a number of low-level tasks and assigns them to individual sub-systems. Furthermore, these sub-systems are organized together through a single experiment control sub-system that is able to orchestrate their operations into a single co-ordinated activity. While such a design has been proven to be both flexible and reliable, when problems do occur, the relatively "loose" connection between subsystems can make proper diagnosis difficult. A well-designed operations log is a key tool in allowing rapid discovery and correction of individual sub-system problems.

All sub-systems will be able to log messages to a single, system wide operations log. Each message will be tagged with the name of the sub-system as well as the time of entry. Each message will also include numerical and text fields to describe error conditions as well as well as global state information such as experiment control run number and state. While there will be provision for one or more free form text areas, the format will include mandatory fixed format fields to facilitate automated searching in one of several modes. At a minimum, the log will be searchable by sub-system, time and global run number. In practice, this operations log may be built on existing open source or commercial database technology.

J.2. DAQ and Online Monitors

- **DAQ and Operations Monitoring**

It is anticipated that, at least for purposes of detector and data acquisition system monitoring, there will be a number of groups interested in having independent and uncoordinated access to one or more software sub-systems at the South Pole. As described earlier, design distinctions between monitoring requests and control commands will allow the use of robust, commercial-grade Web server programs for a large portion of these user interactions. Routing requests for monitoring data through a Web server concentrates most of the interactive data flow through a well-debugged program that acts as a buffer between off-site data requests and data acquisition and on-line sub-system programs. Unfortunately, this mechanism is still subject to the periodic and occasionally unscheduled communications outages between the northern hemisphere and the South Pole. Given the wide distribution of local time zones within the Ice Cube collaboration, this can create limited or, at a minimum, inconvenient access to monitoring data.

With the interposition of a northern hemisphere server/cache gateway system, much of the data acquisition and on-line sub-system monitoring data can be made available to collaboration members at their home institutions during reasonable day-time hours. This gateway system would be configured to refresh its cache from sources at the South Pole during times of expected communications availability and serve these monitoring data files in response to browser and client requests. During times of communications blackout between the South Pole and northern hemisphere, collaboration members would be able to direct their browsers and client programs to the gateway system and obtain current, within 24 hrs., detector and sub-system status. Furthermore, if it were deemed necessary to restrict general collaboration access to South Pole systems, this architecture would provide a straight forward mechanism for distributing detector monitoring information. Thus, up-to-date detector and data acquisition system performance data is made available collaboration-wide on a best effort basis.

- **Online and Offline Monitoring Needs**

Portions of the monitoring and operational information will be needed for correct interpretation of data during later analysis. This can include current and cumulative trigger rates for all trigger types as well as event builder statistics and gross DOM data rates on a string by string basis. There must be a mechanism for identifying and passing monitoring data that should be passed from the DAQ system to the Online system. This may consist of a separate data stream, or even the periodic inclusion of monitoring data in normal data streams.

K. Online Components

The Online system is essentially a carefully organized portion of the Offline environment that has been targeted for execution at the data acquisition site under the control of the experiment control sub system. It has three primary responsibilities. First, it is the sole component responsible for copying the DAQ event stream from the shared disk system, located on the Online LAN, to both archive storage (i.e. tape) and, via satellite, to the northern hemisphere. As such, it is capable of reformatting and repackaging data from the event builder into those forms best consumed by offline analysis tasks. Secondly, it provides the framework and control mechanisms for executing any online filter operations used to analyze, reduce or reject data from any of the data streams emitted by event builders. And thirdly, as described above, the Online system contains filters and or programs that perform all calibration functions beyond those of DOM time base synchronization performed by the DAQ system

L. DAQ Computing Environment

L.1. COTS Hardware and Software Components

There has been a real effort, during the design of both the DAQ and Online systems, to maximize the potential for using commercial, off the shelf hardware and software components (COTS). At a minimum, this includes use of open source operating systems (namely, Linux), standards-based network and communications software and PC-based computing platforms. But, we feel it also should allow for the purchase of standards-based components that have real added value for IceCube's operation. This can, for example, include purchase of PC clusters that have well engineered cooling and maintenance features or disk systems that have extensive space management and diagnostic software packages.

L.2. String LAN Components

As noted earlier, each string terminates in its own private network. In practice, this network consists of a set of DOM hubs and a single processor connected to an off-the-shelf switched Ethernet hub. This design nicely encapsulates the entire string behind each string processor, which serves as the complete interface to the string. Based on system performance, we may chose to have two or more strings share the same string processor. If so, we need only increase the number of Ethernet attachments required for the commercial switched hub.

L.3. String, Global Trigger, and Event Builder Processors

Depending upon the degree to which multiple OM strings can be accommodated on a single string LAN and processor, the IceCube DAQ, when fully implemented, will have on the order of 50 to 100 processors. By any measure, this is a substantial number of systems to house and maintain. When designing a system with such a large number of processors, several factors besides basic unit processor cost must be considered. Many vendors have addressed the ancillary issues involved in configuring, operating, and maintaining large clusters of PC-based Linux systems-most notably for use in the e-commerce industry. For these markets, vendors have addressed such issues as increased packing density (e.g. one processor per 1U rack space), improved ventilation and temperature monitoring, improved, tool-free, modular repair capabilities, and network based remote diagnosis and control of individual processors. These systems typically have a full, but limited set of disk, communications, and PCI resources. In determining the functionality and interconnections needed to implement both the DAQ and Online systems, care has been taken to stay within the physical and logical constraints presented by these PC cluster implementations.

L.4. Event LAN

The data rates presently anticipated indicate that the communications network used to connect all OM string processors with the single global trigger processor and multiple event builder processors can be adequately serviced by conventional data network (100BT or Gigabit Ethernet). As with the String LANs, this network is well bounded and only requires a local switch-based infrastructure.

If these data rate projections grow beyond the capabilities of either 100BT or Gigabit Ethernet technologies, alternate, off the shelf solutions can be inserted into this architecture without major perturbations to the system design. In particular, we have investigated the Myricom switched fabric inter-processor network as a potential solution.

L.5. Storage Systems and Online LAN

As with network infrastructure, storage systems represent an area where system data rates dictate the technical solution. The primary storage facility shown in this design is that which provides the flexible data buffer between the DAQ and Online system partitions. This system design effectively increases performance by providing parallel computing resources for those paths with the highest data throughput rates. At the juncture of the DAQ and Online systems, this implies a shared network based storage system with multiple access paths to both event builder and online processors. The current aggregate data rates indicate that a 100BT based commercial SAN (Storage Area Network) system will meet the necessary storage and throughput requirements. As with the selection of processors, storage system selection should not be based solely on hardware costs. Most modern commercial systems include system level features such as automatic mirroring, local automated backup and remote fault diagnosis and configuration tools. These facilities should receive high consideration in vendor and system selection.

As with the event builder LAN, the technical solutions are sensitive to the ultimate data rates expected within the system. If the current projected data rates prove inadequate, consideration will be given to their effect on the I/O bandwidth requirements of the shared disk system. Alternate solutions with substantially higher throughput rates do exist-most notable fibre channel based SAN arrays. Since these storage systems use individual PCI interfaces and a separate switched fabric interconnect for all disk related data, the costs –as well as the data throughput- increase dramatically.

M. Appendix

Spreadsheet: IceCube Data Rates and Bandwidth Requirements (Preliminary)

Quantities in bold are calculated based on non-boldface quantities.

Description	#	Units	Comments	
basics				
Number of Strings	81			
Number of OMs per String	60			
Total Number of OMs	**4860**			
Dark Noise Rate per "normal" OM	500	Hz		
Dark Noise Rate per "noisy" OM	4000	Hz		
Fraction of noisy OMs	0			
Total Average Noise Rate per OM	**500**	Hz		
Fraction of Hits Needing Waveforms	0.05		less than 0.1, more than 0.02, based on preliminary study of DOM waveforms	
Bytes per Waveform Hit	208		3 channels * 10bits/samp * 128 samples = 480 bytes. Must add 6 to 9 bytes for time stamp. Not all channels are needed, and compression could significantly lower the required number of bytes.	
Bytes per non-waveform Hit	9			
Avg. bytes per hit	**18.95**	bytes		
Data Rate for a single DOM	**9.25**	kB/sec	**92.5**	**kbps, with 8b10b encoding**
Data Rate for a DOM pair	**18.51**	kB/sec	**185.1**	**kbps, with 8b10b encoding**
Data Rate for a string	**555**	kB/sec		
Total Data Rate for noise Hits in Icecube	**44,000**	kB/sec	(the current plan is NOT to save all these hits!)	
string coincidences				
Number of DOMs per DOMHub	6			
Number of DOMHubs per String Processor (SP)	10			
Number of DOMs per SP	**60**			
Number of String Processors	81			
Rate of Filtered, Coincidence Events on a String	60	Hz		
Average Number of Hits per Coincidence	4			
Rate of Hits in Coincidence, per string processor	**240**	Hz		
Number of Bytes per String Trigger Message	**36**	bytes		
Data Rate for String Triggers, per String Processor	**2160**	bytes/sec		
Total Data Rate for String Processors	**170.9**	kB/sec		
global event triggers				
Gobal event rate	1000	Hz		

Size of global event message	10	bytes	8 bits - trigger type 50 bits - trigger time 8 bits - pretrigger time interval 8 bits - posttrigger time interval 3 bits - target event builder
Number of string processors receiving trigger message	81		
Total data rate for global events	**791.0**	kB/sec	
event lookback messages			
Rate of global events	1000	Hz	
Number of string processors	81		
Trigger window	8000	nsec	
Number of random noise hits per event	**19.44**		probable underestimate - need correct poisson statistics
Number of muon hits per event	12		
Average number of hits/event	**31.44**		
Average number of hits/string/event	**0.4**		
Average bytes/hit	**18.95**		includes waveforms
Average data rate per SP	**7.2**	kB/sec	
Total data rate for hit lookback	**581.82**	kB/sec	